# Testing - When Do I Stop?

Robert E. Waterman,
Unisys Government Systems
10265 Aerospace Drive, Lanham, MD
20706, 301-794-2711

Lawrence E. Hyatt,
NASA Goddard Space Flight Center
Greenbelt, MD 20771
301-286-7475

The Software Assurance Technology Center (SATC) in the Systems Reliability and Safety Office at Goddard Space Flight Center (GSFC) is investigating the use of software error data as an indicator of testing status. Items of interest for determining the status of testing include projections of the number of errors remaining in the software and the expected amount of time to find some percentage of the remaining errors. To project the number of errors remaining in software, one needs an estimate of the total number of errors in the software at the start of testing and a count of the errors found and corrected throughout testing. There are a number of models that reasonably fit the rate at which errors are found in software, the most commonly used is referred to in this paper as the Musa model. This model is not easily applicable at GSFC, however, due to the availability and the quality of the error data.

At GSFC, useful error data is not easy to obtain for projects not in the Software Engineering Laboratory. Of the projects studied by the SATC, only a few had an organized accounting scheme for tracking errors, but they often did not have a consistent format for recording errors. Some projects record errors that were found but did not record any information about resources applied to testing. The error data frequently contained the date of entry of the error data rather than the actual date of error discovery. In order to use traditional models such as the Musa model for estimating the cumulative number of errors, one needs fairly precise data on the time of discovery of errors and the level of resources applied to testing. Real world software projects are generally not very accommodating when it comes to either accuracy or completeness of error data. The models developed by the SATC to perform trending and prediction on error data attempt to compensate for these shortcomings in the quantity and availability of project data.

In order to compensate for the quality of the error data, the SATC developed a software error trending models using two techniques, each based on the basic Musa model, but with the constant in the exponential term replaced by a function of time that describes the 'intensity' of the testing effort. The shape and the parameters for this function can be estimated using measures such as CPU time or staff hours devoted to testing. The first technique involves fitting cumulative error data to the modified Musa model using a least squares fit that is based on gradient methods. This technique requires data on errors found and the number of staff hours devoted to testing each week of the testing activity. The second technique uses a Kalman filter to estimate both the total number of errors in the software and the level of testing being performed. This technique requires error data and initial estimates of the total number of errors and the initial amount of effort applied to testing. Only the first of the two techniques will be discussed

in this paper.

The SATC has currently examined and modeled error data from a limited number of projects. Generally, only the date on which an error was entered into the error tracking system was available, not the date of discovery of the error. No useful data was available on human or computer resources expended for testing. What is needed for the most accurate model is the total time expended for testing, even if the times are approximate. Using the sum of reported times to find/fix individual errors did not produce any reasonable correlation with the resource function required. Some indirect attempts to estimate resource usage, however, led to some very good fits. On one project errors were reported along with the name of the person that found the error. Resource usage for testing was estimated as follows: A person was estimated to be working on the testing effort over a period beginning with the first error that they reported and ending with the last error that they reported. The percentage of time that each person worked during that period was assumed to be an unknown constant that did not differ from person to person. Using this technique led to a resource curve that closely resembled the Rayleigh curve (Figure 1).

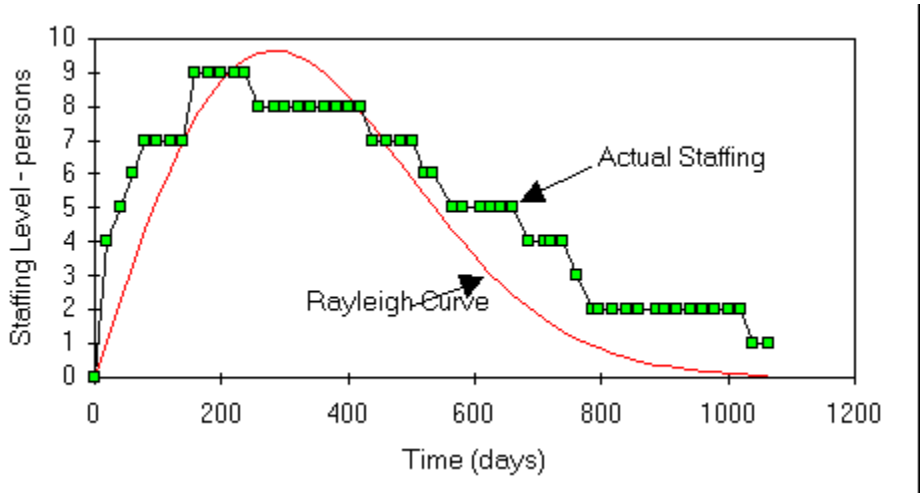


**Figure 1: Test Resource Levels for Project A**

On most of the projects, there was good conformity between the trend model and the reported error data. More importantly, estimates of the total number of errors and the error discovery parameter, made fairly early in the testing activity, seemed to provide reliable indicators of the total number of errors actually found and the time it took to find future errors. Figures 2 shows the relationship between reported errors and the SATC trend model for one project. The graph represents data available at the conclusion of the project. This close fit was also found on other projects when sufficient data was available.

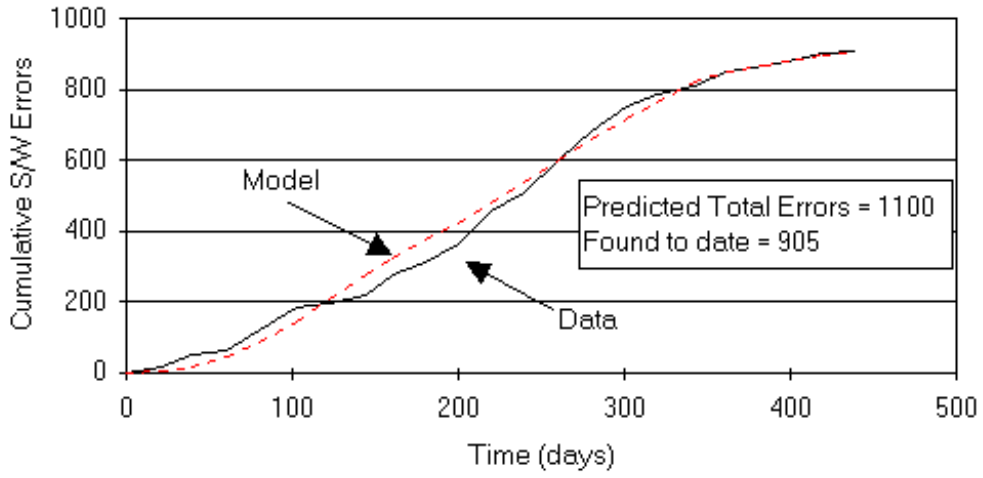


**Figure 2: Cumulative Software Errors for Project A**

On another project, different estimates of the total number of errors were obtained when estimates were made over different testing time intervals. That is, there was inconsistent agreement between the trend model and the error data over different time intervals. Through subsequent discussion with the project manager it was learned that the rate of error reporting by the project went from approximately 100% during integration testing to 40% during acceptance testing. Furthermore, there was a significant amount of code rework, and testing of the software involved a sequential strategy of completely testing a single functional area before proceeding to test the next functional area of the code. Thus, the instability of the estimates of the total errors was a useful indicator of the fact that there was a significant change in either the project's testing and reporting process. Figure 3 shows the results for this project. Note the change in slope of the reported number of errors occurring around 150 days. The data curve flattens at the right end of the curve due to a pause in testing, rather than a lack of error detection. This project is still undergoing testing.

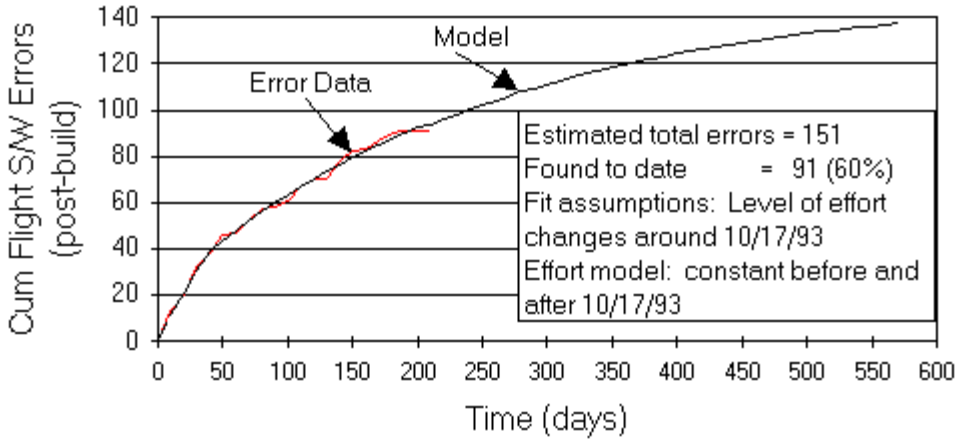


**Figure 3: Cumulative S/W Errors for Project B - Flight S/W**

If error data is broken into the distinct testing phases of the life cycle (e.g., unit, system, integration), the projected error curve using the SATC model closely fits the rate at which errors are found in each phase.

Some points need to be clarified about the SATC error trend model. The formulation of the SATC equation is the direct result of assuming that at any instant of time, the rate of discovery of errors is proportional to the number of errors remaining in the software and to the resources applied to finding errors. Additional conditions needed in order for the SATC trending model to be a valid are:

1. The code being tested is not being substantially altered during the testing process, especially through the addition or rework of large amounts of code.

2. All errors found are reported.

3. All of the software is tested, and testing of the software is uniform throughout the time of the testing activity.

Condition 1 is present to ensure that the total number of errors is a relatively stable number throughout the testing activity. Conditions 2 and 3 are present to ensure that the estimate of the total number of errors is in fact an estimate of the total errors present in the software at the start of testing - no new errors are introduced during testing. If testing is not "uniform" then the rate of error discovery will not necessarily be proportional to the number of errors remaining in the software and so the equation will not be an appropriate model for errors found. No attempt will be made here to make precise the meaning of the word "uniform".

The SATC developed this model rather than using the standard Musa model because it seems less sensitive to data inaccuracy and provides for non-constant testing resource levels. An additional benefit from this work is the application of the Rayleigh curve for estimating resource usage. Future work by the SATC will continue to seek a practical balance between available trend analysis theory and the constraints imposed by the limited accuracy and availability of data from real-world projects.